VŠB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

Individual Professional Practice in the Company

Absolvování individuální odborné praxe

2019 Ondřej Rusz

VŠB - Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

Bachelor Thesis Assignment

Student:	**Ondřej Rusz**
Study Programme:	B2647 Information and Communication Technology
Study Branch:	2612R025 Computer Science and Technology
Title:	Individual Professional Practice in the Company Absolvování individuální odborné praxe
The thesis language:	English

Description:

1. The student shall undergo individual practice with company Y Soft Corporation, a.s.
2. The final report structure shall be the following:

a/ Specification of the professional orientation of the firm where the student underwent his/her professional practice, specification of his/her occupational category.
b/ The list of tasks the student was assigned in the course of his/her professional practice including their time spans.
c/ Methods chosen when dealing with the given tasks.
d/ Theoretical and practical knowledge and skills acquired in the course of his/her professional practice.
e/ Knowledge and skills the student were missing in the course of his/her professional practice.
f/ Results achieved in the course of his/her professional practice and the general assessment of them.

References:

In accordance with the tutor guiding the student's professional practice.

Extent and terms of a thesis are specified in directions for its elaboration that are opened to the public on the web sites of the faculty.

Supervisor: **Ing. Lenka Skanderová, Ph.D.**

Consultant: Ing. et Ing. Jakub Pavlák

Date of issue: 01.09.2018
Date of submission: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
Head of Department

prof. Ing. Pavel Brandštetter, CSc.
Dean

I hereby declare that this master's thesis was written by myself. I have quoted all the references I have drawn upon.

Ostrava, 29 April 2019

I hereby agree to the publishing of the bachelor's thesis as per s. 26, ss. 9 of the Study and Examination Regulations for Bachelor's Degree Programmes at VŠB – Technical University of Ostrava.

Ostrava, April 29, 2019 .....

I would like to thank my consultant Ing. Lenka Skanderová, Ph. D. for swift and valuable remarks and for the time spent on the consultations. I would also like to thank my girlfriend, family, and colleagues.

Abstrakt

Cílem této bakalářské práce je popsat mé působení, přínos a získané znalosti ve společnosti Y Soft Corporation, a.s. Během praxe jsem se věnoval testování robotickou rukou v rámci robotického týmu. Za pomocí programovacího jazyku Python a Robot Framerwoku jsem testoval software do tiskáren SafeQ, který je mimo jiné nasazený i na VŠB. Dále jsem se věnoval optimalizaci databázových dat pro rozpoznávání obrazovek, definici tlačítek a sekvencí příkazů, používaných k práci s robotem.

Klíčová slova: automatizace, testování, praxe, Python, Robot Framework, Y Soft Corporation a.s., bakalářská práce

Abstract

The goal of this bachelor thesis is to delineate my performance, benefit and acquired knowledge in Y Soft Corporation, a.s. During my practice, I focused on robotic arm testing within the robotic team. Using the programming language Python and Robot Framework I tested software for printers called SafeQ, which is also deployed at VŠB. Furthermore, I worked on optimizing database data for screen recognition, button definitions and instruction sequences utilized by the robot.

Key Words: automation, testing, practice, Python, Robot Framework, Y Soft Corporation a.s., bachelor thesis

Contents

List of symbols and abbreviations

RQA	–	Robotic Quality Assurance
RMS	–	RQA Management System
RF	–	Robot Framework
IMPROC	–	Image Processing

List of Figures

List of Tables

Code Listings

1 Introduction

The goal of my bachelor thesis was to perform a technical practice in company Y Soft Corporation, a.s., specializing in large-scale printing solutions using its software called YSoft SafeQ. The reason I chose to do a bachelor practice was to acquire knowledge about how software development takes place in a large group and to find out what it takes to be a part of it. Another reason was to apply the theoretical skills acquired during my studies at the university into real life and therefore bring value to the company.

I was hired based on an interview verifying my knowledge and my eligibility to carry out the work I would be later assigned. I was a part of Robot Team which develops custom software for a robotic arm used to automate regression testing on printers.

This final thesis consists of four chapters. In the first one, I will describe the company and the team I was working with. The next chapter presents used software and technologies, programming languages, frameworks, custom company-created tools, and third-party software. The third chapter describes tasks I had to handle during my practice, problems I had to overcome and the workflow I used to do so. The last chapter concludes my practice and summarizes acquired knowledge and new skills.

2 Y Soft Corporation, a.s.

2.1 About the company

Y Soft Corporation, a.s. is a software and hardware company established in the year 2000 in Brno. It currently operates in 17 countries around the world with over 300 employees. Y Soft has over 14,000 customers and its solutions are used in 121 countries across 6 continents. The company's main focus is to reduce print-related costs, increase its effectiveness and enhance security. YSoft SafeQ is a specialized software designed to reach this goal and is aimed at medium and large businesses.

Y Soft also develops its proprietary hardware used to enhance printers with additional functionality. These solutions include USB Card Reader and SafeQ Terminal.

2.2 Team

I was included in Brno's Robot Team. Robot Team oversees the development and improvement of a whole system built around a custom robotic arm used for testing multifunction printers. Robot Team is one of the smaller teams in Y Soft with around 13 members, most of whom are students working on their theses.

Team's workflow is organized into fourteen day long sprints. Within this time span, the team members are set to complete a predetermined amount of work. Every day there is a stand-up meeting to keep everyone informed about the current progress of the assigned tasks. At the end of each sprint, the team has a meeting to look back at the last sprint and plan the next one. These agile focused methods helped me primarily at the beginning of my practice when I had to familiarize myself with the product and the workflow inside of the company.

3 Used software and hardware

3.1 YSoft SafeQ platform

YSoft SafeQ is a Managed Printing Services software which helps users to manage, optimize and secure printing and scanning.

These are its main features:

- **Pull printing**

  Pull printing is a feature where a user sends their print job to a dedicated server rather than directly to a printer. Therefore the job can be released on any device connected to the network.

- **Secure access**

  Another platform's advantage is related to pull printing and enables an extra layer of security. Logging into each device allows the user to access their print jobs and other features available to them. Secure access eliminates the danger of confidential documents falling into the hands of an unauthorized person because print jobs can only be released from the device's terminal after logging in.

- **Scan workflows**

  These XML templates describe scanning workflow step by step, define the basic settings, output file and the workflow parameters themselves. They enable user-friendly, one click scanning with predefined settings, for example, scan to e-mail or scan to a network folder.

- **Accounting**

  Accounting is a feature whose objective is to accurately track different aspects of print related services.

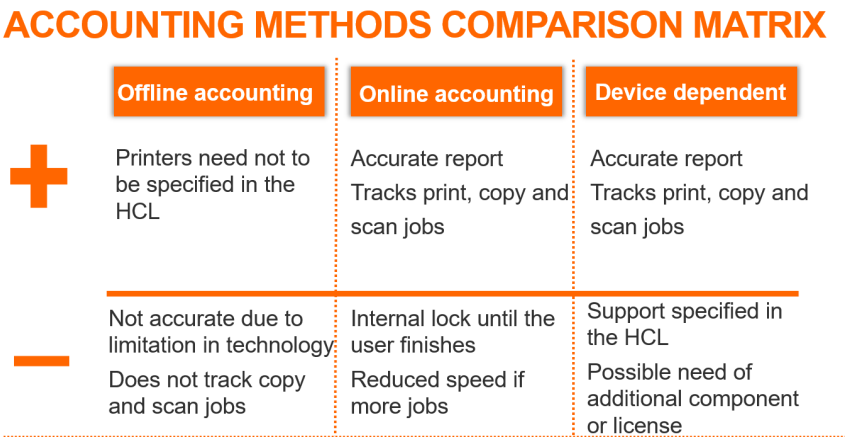

Figure 1: Accounting Methods [1]

SafeQ can use three types of accounting methods mentioned in Figure 1:

- *Offline accounting* monitors every print routed to a printer via the SafeQ server in real-time before it is actually printed. Therefore its accuracy is not perfect because it does not take the device status or the actual printer output into account. For example, a color page printed on a black&white printer is still accounted as a color page.
- *Online accounting* is a more accurate method of monitoring activity on printers. The main difference is waiting for the job to be accounted after the printing is complete. This method requires an average latency of less than 200ms between the device and the server.
- *Device dependent accounting* tracks the number of pages that have actually been printed by the printer. Even though this is the most accurate method, the accuracy is still not flawless and is between 95 and 98 percent. This is caused by special types of print jobs such as printer status page or pages printed from external hard drive bypassing the SafeQ application.

3.2 Robot Framework

Robot Framework [2] is an open source framework used for acceptance testing. Its syntax is easy to read and can be extended by libraries implemented in Python, Java or .NET. Test data can be specified using the following command line options:

- `-test (-t)` runs a test specified by a test case name or a pattern supporting logical operators and simple regex like expressions.
- `-suite (-s)` executes a whole test suite by its name or pattern
- `-include (-i)` and `-exclude (-e)` keywords used to select tests by their tags.

Each test case may be labeled with one or more tags which are used to bundle test cases from different test suites together. Tags can also be forced on a whole suite using the 'Force Tags' keyword, projecting them on every test case within the test suite. Bundling is a very efficient way to run tests without any human interaction apart from starting the test itself. After the set of tests finishes, a report is created for debugging assuming that some tests fail.

3.3 Python

Python [3] is a high-level, object oriented, dynamically typed programming language created by Guido van Rossum in 1990. In our project, Python was chosen to extend Robot Framework with custom add-on libraries. It is a language I was familiar with from the university and I could use my knowledge to read and understand the code comfortably.

3.4 RMS Web Interface

RMS web user interface is one of the vital parts of the robotic system. It is used to define reference screens, buttons placements and flows, calibrate robot's camera, manage reports, trigger training of AI for image processing and also provides statistics about the test coverage for each vendor. My job was to use this UI from the user's point of view. I had to use all of the tools for the regression tests, find bugs, analyze the web from the user experience standpoint and share my feedback with the team.

- **Reference screens**

  Snapshots of device's screens captured by the camera which are categorized by device model, screen name and terminal version. These references are later used to train RAINOC (see subsection 3.5) in order to accurately recognize the reference screen which the robot's camera is currently looking at.

- **Button placements**

  Buttons carry information about the functions of a specific element on a screen. Buttons can be navigational – causes the screen to change on tap (e.g. login); or functional – changes something within the same screen (e.g. keyboard input keys). Each button is tied to one or more reference screens and can also have more than one behavior, meaning it can be set to navigate between several screen pairs.

- **Flows**

  Flow is a sequence of buttons or fictive actions, for example, a swipe of a card, which is performed by a tag emulator. Each step can be optionally followed by verifying of the button's destination screen. This verification is mainly used on the final step of the flow to ensure its correctness. Flow is tied to a specific device and terminal type which guarantees it being used during an appropriate test case.

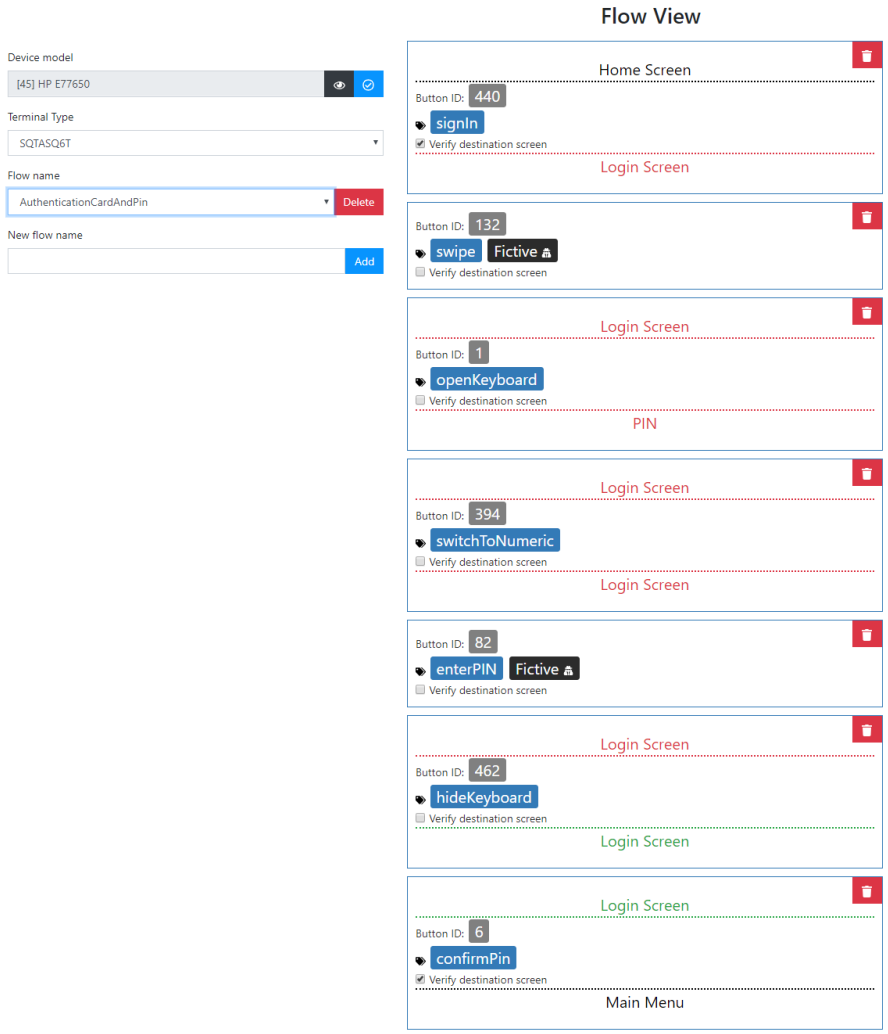

Figure 2: Authentication using a card and PIN

The flow in Figure 2 describes the sequence of buttons used for authentication using a card and a PIN. The robot navigates to Login Screen by tapping the appropriate button, swipes a card and inputs the PIN using a sequence of buttons. Flow ends by confirming the PIN and checking whether robot navigated successfully to the Main Menu screen.

- **Feature regions**

  Feature region is a region of a reference screen used to perform specific actions during tests. Its original purpose was to manually define important features of a screen. The naïve (deprecated) image processing would then recognize the screen based on these elements. This was not a reliable approach especially with large data sets consisting of dozens of reference screens and caused irregularities in image recognition by inaccurately classifying screens. Currently, feature regions are only used for reading text in a specific location, for example, a name of a print job.

3.5 RQA GUI and IMPROC

These two computer programs are vital in communicating with the robotic arm and the camera. Robotic Quality Assurance Graphical User Interface, or RQA GUI for short, handles communication with the robot and is used for calibrating and moving the robot to the desired position. It works together with IMPROC, which handles image processing from a camera and feeds information to the GUI for it to act accordingly. Image processing used to work using a naïve algorithm, which was for its unreliability replaced with a new approach called RAINOC.

RAINOC (Robotic Artificial Intelligence Objective Classifying) is an image recognition software powered by neural networks. It sped up image recognition by more than 80% and made it significantly more accurate. It also eliminated the need to manually define feature regions as artificial intelligence can determine the screen's features automatically.

3.6 Atlassian tools

Atlassian is an Australian company providing enterprise software for developers and teams. I used several of their tools in order to track tasks, version my code and to deploy plans outside of work time.

The used tools comprise of:

- **Sourcetree [4]**

  The Git client I used was Sourcetree. The reason I chose it was because of its user-friendly interface and its compatibility with the company's repository storage of choice called Stash. It was my first experience with software versioning in a bigger team. I had to learn how to categorize code commits into correct branches corresponding to tasks in JIRA.

- **JIRA [5]**

  JIRA is a software used for project management and bug tracking. It takes care of the current sprint board and assigned tasks or issues and facilitates time tracking.

- **Bamboo [6]**

  Bamboo is a deployment server used to create multi-stage build plans. It supports adding time-based triggers which can be set to run plans outside of work time. It simplifies continuous integration by running automated tests.

4 Tasks assigned during practice

The main goals of my work in the company were:

1. **Automate testing using the robotic system**

   I had to create test cases utilizing a robotic hand to test SafeQ via terminals on printers. I was also the only detached tester of the whole robotic system and the interface outside of the team's headquarters, so I was to provide feedback and ideas about the RMS Web Interface. I worked on improving automated test coverage and toward its full automation.

2. **Manage said tests and validating their stability**

   I bundled these tests into test plans, which would be testing the latest build of SafeQ every night. I had to manage them because of the nature of robotic testing. Unlike software-only based testing, robotic tests are more prone to failure due to various external actions. This includes problems on printers (usually solved by a simple restart), unintentionally moving the robot or camera, rendering them badly calibrated or the robot controlling computer suddenly deciding to update. The SafeQ's UI also may change slightly, requiring a fix in tests as well.

3. **Expand test coverage**

   I had to expand and maintain the central reference screen database. This includes checking whether the screens are well organized, covering more devices by expanding the database with new data and working with button placements. The correctness is checked by running plans and adjusting the data if needed.

4. **Save time**

   The above-mentioned goals were ultimately aimed at saving time. Manual testing is a tedious job and would waste a lot of time every sprint. Since the tests are almost the same every time, the robotic system is being developed to transfer this workload on robots.

4.1 Authentication

The first introductory task was aimed mainly to acquire knowledge about the whole robotic system. I had to automate every supported type of authentication on an HP device. Testing with a robot can be split into several steps which must be performed in order.

The first step is getting the hardware ready. We need a robotic arm and a camera on tripods and a tested device as well. The robot has to be placed in front of the device's terminal so it can reach every corner of the touch screen and other buttons around it. The camera has to be placed facing downwards from above in such a way that its vision does not get obscured by the robot and has a clear view of the terminal screen. Afterward, both camera and robot have to be calibrated. The robot is calibrated manually using the RQA GUI software by navigating it

into 3 corners of the screen, collecting the respective coordinates. The robot is navigated using keys W, A for X coordinate, A, D for Y coordinate, R, F for Z coordinate and Q, E for stylus tilt. The stylus should tap the screen perpendicularly to prevent sliding.

The rest of the screen is translated based on the corner points in space and the robot is therefore calibrated. The camera's calibration is carried out in RMS and has two options — automatic or manual. Automatic utilizes a device-specific paper marker placed around the edge of the display. The manual one works by simply selecting all four corners of a screen on a still image captured by the camera.

SafeQ enables secure access to the printer not only via screen input methods such as PIN or username and password but also supports logging in using a personal card. Most devices do not have a card reader so a proprietary USB connected card reader is used. A tag emulator is a device which is able to generate 16 digit hexadecimal codes. It is placed on the reader in order to emulate a card swipe.

The second step is defining all the required reference screens. Reference screen is a snapshot from the camera assigned with a name combined with terminal type. Authentication tests are the simplest ones and usually only need three screens:

- **Home Screen** — the first screen the device shows after re-installation
- **Login Screen** — the screen which shows authentication options and possible text fields to enter credentials
- **Main Menu** — the first screen after authenticating successfully

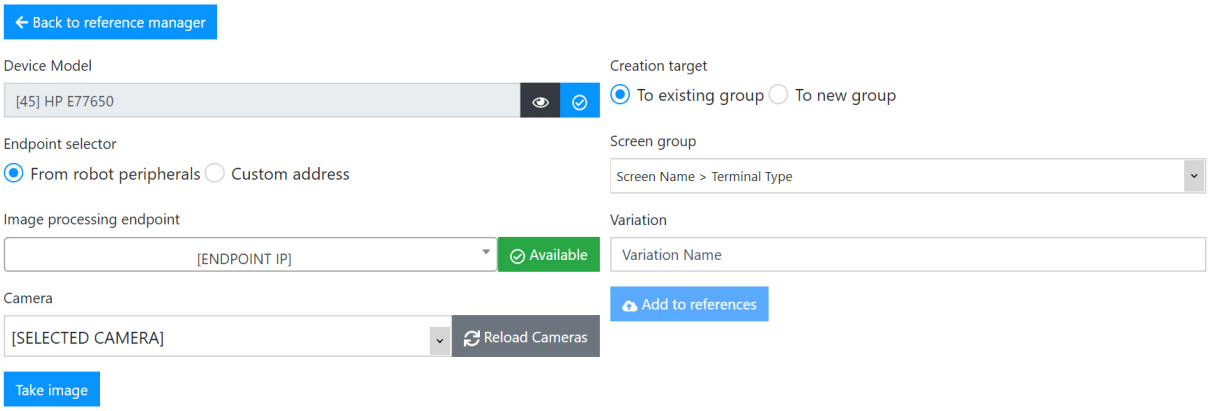

Figure 3: Creating a reference from an image captured by a camera

Reference screens are defined using a tool in the RMS (see Figure 3). A new reference screen can be added either by snapping a photo from a camera or by uploading a file. The new snapshot needs to be categorized by a specific Device Model combined with a Screen Group, which consists of the screen and the terminal type the device terminal has installed. After the reference screens are added into the database, RAINOC has to be triggered to train itself to

recognize them. This process does not take long but requires properly categorized, clear and focused reference screens images.

If there are two very similar screens, image recognition might still incorrectly evaluate them and RAINOC has a failsafe for this kind of situations. It is possible to define screen distinguishers for a pair of screens. They are used when one of the defined screens is recognized and second processing is performed, focusing on manually selected screen elements. With this precaution and reference screens correctly defined, the image processing closes to 100% accuracy.

Next step is specifying button placements on each of the screens using the visual editor in the RMS. We need to correctly link the screens together so the robotic system has a way to navigate between them. If we need to navigate to a specific screen during a test procedure, the system can find the best path based on these links and we don't need to specify each button tap individually.

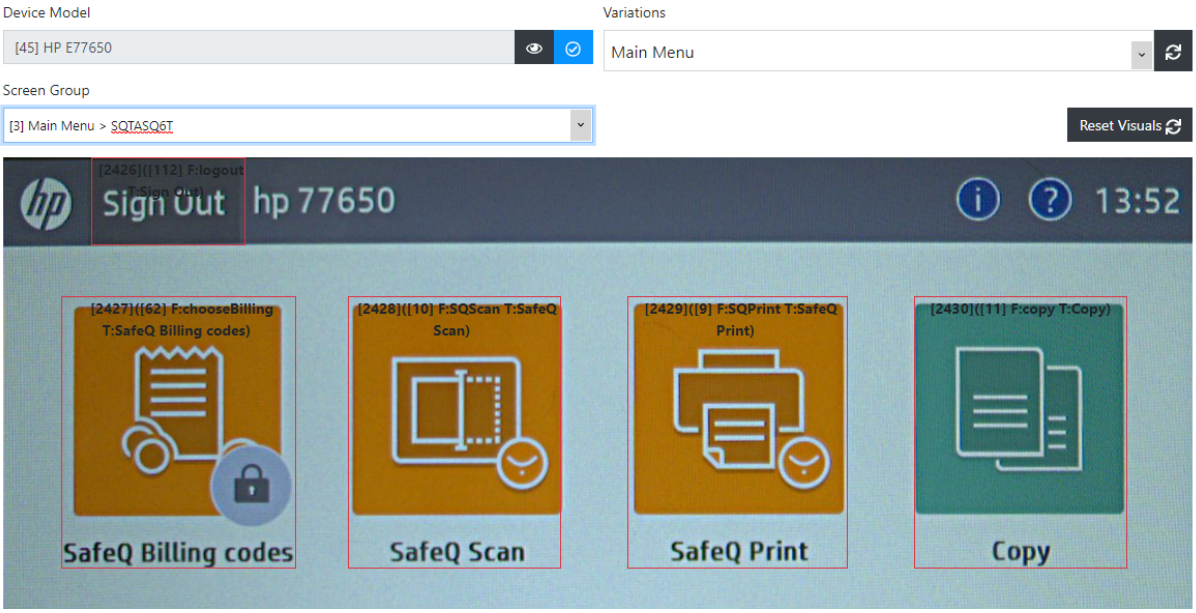

Figure 4: Button placements on a Main Menu Screen

Each button can be assigned to one or more screen groups. Figure 4 shows how a screen with defined buttons looks. The applications might sometimes install in a different order so each of the buttons is set to first try to find the text associated with the button instead of blindly pressing the marked position.

The robot needs to know the difference between an authenticated and unauthenticated screen, in most cases authentication procedure navigates from Login Screen to Main Menu by inputting correct credentials or providing an emulated card swipe to the card reader. There are three different authentication methods: card, PIN or username and password. In practice, these methods are often combined and SafeQ supports almost every combination. This resulted in a total of 10 test suites with 37 test scenarios.

The next important step is to set up the global RF variables used to determine which device (vendor name, ID and the device's network IP), robot (its ID and IPs for RQA GUI and IMPROC) and SafeQ server IP is used. These variables are stored in a standalone file 'robot.config' which is accessible from the whole project.

The last step is to run the actual tests using Robot Framework. I used Gherkin-like syntax handling the test logic using `Given`, `When` and `Then` keywords paired with Robot Framework custom keywords designed for testing SafeQ. This approach provides a human-readable format, encapsulating the python back end code.

Before the actual testing begins, RF runs a Suite Setup which prepares the environment and a Test Setup which prepares the robot and the device. After the device is reinstalled with the desired authentication method, every test starts by creating a user on the SafeQ server. This is a precondition for the whole test and uses the keyword `Given` followed by `Add New User`. The function to add a user generates a unique username, password, PIN, card code and stores the data in a dictionary which is then posted to the SafeQ server via an HTTP request.

Table 1: Authentication test suites

Test Suite Name	Number of tests
Card	3
PIN	2
Username and Password	2
Card and Conditional PIN	6
Card and PIN or Username and Password	6
Card and PIN	4
Card and Username password	4
Card or PIN	5
Card or Username password	5
Total	37

The test suites combine correct and incorrect credentials to find out whether the authentication system behaves as expected in every case and not only when correct credentials are entered.

I implemented a total of 37 tests covering every possible authentication method supported by HP. The mentioned methods are split between the test suites mentioned in Table 1. Each of these suites consists of several test cases covering both valid and invalid credentials.

In the following list, I would like to go into detail about the test suites mentioned in Table 1

- **Card suite** contains three tests. One using a valid card, one using invalid and then valid card and the last using a card to log out.
- **PIN suite** has two tests. The first one checking a valid PIN, the second one first tries an invalid PIN and then a valid one.
- **Username and Password suite** is similar to the PIN suite, only working with username and password instead of PINs.
- **Card and Conditional PIN** is a special type of authentication method because it can be different depending on a user. Some users can be required to use both a card and then a PIN, but for others, just a card can be sufficient. The six test cases cover all combinations for these two different users.
- **Card and PIN or Username and Password suite** is a more complex suite utilizing more than one authentication method. This security measure allows to either log in with a card, followed by entering a PIN; or just by username and password. This suite needs to cover all combinations of these methods so it consists of correct card and PIN, then two more test cases, where first card and then PIN are incorrect. The next test case checks behavior if both of these methods are incorrect and card and PIN part is covered. Next we only need correct username and password, and lastly incorrect username and password followed by a correct one.
- **Card and PIN suite** is a subset of the previous suite containing the first four test cases. I had to adjust these tests to work on a slightly different environment since the user interface is different for each of the authentication methods.
- **Card and Username password suite** is another suite combining different sign-in methods. Both of these credentials have to be correct in order to log in successfully. I designed the tests accordingly and the suite consists of four test cases checking all possible combinations of correct and incorrect credentials.
- **Card or PIN suite** is the first suite working with or operator. This means either a card or PIN can be used to log into an account. This leads to new test cases, where the first (incorrect) method can be different from the second (correct) one. This suite consists of two test cases, which are valid right away — correct card, correct PIN. Then incorrect card and correct PIN, and a correct card with an incorrect PIN. The authentication happens after it is submitted, so even though one of the methods is incorrect, the user should still be able to log in via the correct method. The last test case is an unsuccessful login with both methods incorrect.

- **Card or Username password** is a similar suite to Card or PIN. Again needing adjustments on the robot's side, tweaking the button placements and recognizing the different user interface.

Later during my practice, I adjusted these tests to work on a Xerox device as well. The main workflow difference between these devices is to deal with long terminal reinstallation time of the Xerox I worked with. This could be worked around by adjusting the Suite Setup and Teardown to not reinstall the device every time and rather work with a manually installed terminal.

The whole test set can be run from Bamboo using a test plan I created. During this task, I acquired fundamental knowledge about the robotic system which would be later applied in order to solve more complicated assignments.

```
Authentication with 2 methods (OR) - 1 incorrect
    [Arguments] ${method_incorrect} ${method_correct}
    Given User exists in SafeQ
    When Run keyword Robot fails to authenticate with invalid
        ${method_incorrect}
    And Run keyword Robot authenticates with ${method_correct}
    And Robot logs out
    Then Terminal access log exists
```

Code Listing 1: RF test case for authenticating with 2 methods, where the first one is incorrect

Code Listing 1 shows a Robot Framework test case used to perform a test utilizing two authentication methods. The second one of them has to end with on an authenticated screen in order for the test to be successful. An example of calling this test case would be `Authentication with 2 methods (OR) - 1 incorrect PIN Card`. After a user is prepared, a random PIN code is generated assuring incorrect PIN. The robot then tries to log in three times (to check for false negatives) and expects an error. After the robot fails to authenticate with the randomly generated PIN, it can continue with the correct method authentication. The robot authenticates with the correct method, logs out, checks SafeQ reports for terminal log access and if there is an access log for the desired user, the test can be evaluated as passed.

This test case can also be used to test the same authentication method twice, once with incorrect and once with correct credentials. The correct PIN (card code, username and password) is stored in a global variable which is not overwritten during the generation of incorrect credentials. After the robot fails to authenticate, the correct PIN (card code, username and password) is set back.

4.2 Regression Testing

Regression testing is performed to verify whether a software product works as intended after deploying new changes into it. The main task I handled during my practice was the creation and maintenance of automated regression tests for printers.

Regression testing checks the functionality of the device and is designed to detect bugs and irregularities across the platform after each new monthly release. This testing is mostly done manually and automating this process is essential to improve the quality of the product. My main job was to write, adapt and improve automation for individual devices.

These tests use either PIN or Card authentication methods because they are the fastest ones. Since the aim is to test other aspects of the platform, the necessary step of logging in has to be completed as fast as possible. Card authentication is preferred but in the case of a tag emulator defect, PIN authentication is the second fastest method.

In total, I made around 35 tests operational and I would like to focus on five main ones which would later be used as the most stable ones in Nightly Plans (subsection 4.3).

- **Print all with accounting** tests two of the most used features — printing and accounting.

```
Print all with accounting
    [Tags]    HP_2ndGen    Xerox_1stGen    Xerox_2ndGen
    Given User exists in SafeQ
    And User has jobs in secure queue
    When Robot authenticates with print all enabled
    And Robot logs out
    And Jobs have expected status in reports
    And Job has been accounted    pages=${TOTAL_PAGES}
        price=${JOB_PRICE}
```

Code Listing 2: Print all with accounting Robot Framework code

Given the user exists in SafeQ, RF pushes two print jobs on the server. Jobs used in these test scenarios are simple PDF files containing only one line of text in order to save ink or toner cartridge. RF then makes sure jobs are ready on the server and authentication using robot begins. During the authentication procedure, SafeQ has a possibility to choose a `Print all` option which automatically prints all waiting jobs in the queue of the user who is logging in. The robot does not need to do any other action so it logs out and after RF validates the desired jobs were printed and accounted, the test can be finished as successful. I set the total pages and job price to be corresponding to the number and type of pages printed.

I designed this test to check the functionality of SafeQ's reporting and accounting components. It also validates the correct behavior of the print all option.

The biggest problem I encountered with this test was a problem with accounting of a specific Xerox device I worked with. An incorrect file was sent to be printed and although the print itself worked fine, the device could not account the job correctly causing the test to fail.

Another problem happened on an HP device during the selection of the print all option. The robot was too fast and emulated a card swipe right after tapping the print all switch. The switch needs to finish its animation before it is truly selected, so I added a small delay in order for the test to work correctly.

- **Secure print with accounting** is a similar test but expands functionality by adding quotas. Quotas are used to keep track of the amount of jobs users print, copy or scan. Each user can also be limited and after depleting their quota, they can be asked to either recharge their credit for following jobs or wait for a reset of their quota.

  This test adds a quota for a specific amount to the user. It is then depleted by the print jobs. Its progress is extended by checking the user's wallet balance which should be at zero after the job is printed.

- **Direct print with accounting** verifies a special kind of printing queues called direct queue. Direct queue is used to print document right away without the need to log in, therefore, the security is very low and is usually only used in small scale businesses. Documents printed this way should not be confidential. SafeQ can still track, report and account printed jobs and the test analyses these aspects like in previous tests.

- **Scan to folder with accounting** is a test validating the functionality of scan workflows.

```
Scan to folder with accounting
    Given User exists in SafeQ
    And Scan to folder robot workflow is imported
    When Robot authenticates
    And Robot scans 1 page using first workflow
    And Robot logs out
    Then Scan job has status scan and is accounted
         price=${SCAN_PRICE}
    And Scanned file exists
    And Job has been charged    job_price=${SCAN_PRICE}
```

Code Listing 3: Print all with accounting Robot Framework code

An XML file is imported into the SafeQ. The workflow has to be activated and the user is granted access to it. Server services are restarted to ensure correct update of said operations and the robot can start testing.

After logging in, the robot navigates to SafeQ Scan application and selects the imported one. The workflow is designed to scan the document into a network folder so apart from checking reports and accounting in the SafeQ, the test also ensures its validity by accessing this folder and confirming the document has been scanned and uploaded correctly.

Unfortunately, I found out this test is not stable on some devices because their scanner might not be initialized before the cover is lifted. This behavior is device specific and is based on regular usage of devices when every scanning operation is preceded by lifting the cover and inserting the document to be scanned — a step which is missing from the robotic tests. This limitation might be solved in the future by adding a versatile motorized tool which would operate the cover sensor causing the scanner to be initialized on demand without human interaction.

- **Copy with accounting** verifies the printer's copying capability. This test was chosen because it covers the device's native application used for copying files. After user logs into their account, SafeQ employs the device's proprietary application but can track what the device is doing and correctly account for the copy jobs.

4.3 Nightly Plans

The biggest strength of automated robotic testing is its tirelessness. Since a robot can be used outside of working hours it can save time which is otherwise spent testing the platform manually. It is also a great way to ensure healthiness of the system during continuous integration.

This is where nightly plans come in. I designed these plans to run a reduced set of tests every night on the newest snapshot of SafeQ in order to validate immediate changes into the system. The plans consist of the most stable tests mentioned in subsection 4.2 and should yield the least amount of false negatives. My team and I run the plans on most devices supported by SafeQ using Bamboo, which collects results about each run and developers can immediately see whether a change in the system caused any serious problems.

4.4 Knowledge Sharing

At the end of my practice, I conducted several knowledge sharing sessions with my colleagues. I informed them about any changes and new features in robotic testing, basic knowledge about how to operate the robot and what to focus on.

These sessions took place in groups of two to ensure individual approach and to enlighten any question they might have. I divided them into two parts:

- **Theoretical knowledge** where I covered basic steps to perform a test and how to set up the necessary repositories and programs. I went over the current version of RMS they were not yet familiar with and talked about important changes in the workflow. The participants learned about problems they might encounter and how to tackle them.

- **Practical knowledge** used to test the theoretical knowledge at the actual robotic arm. The participants learned how to calibrate the robot and the camera accurately, grasped basic knowledge about how to teach the robot to recognize screens and run basic test scenarios.

The main objective was to show my colleagues how far the robotic project has come but also point out any limitations it might have and how to troubleshoot them.

5 Conclusion

During my practice, I utilized the knowledge acquired at the VŠB-TUO university. The main subjects I benefited from are the ones focused on programming. Scripting Programming Languages and their Applications helped me the most since I used Python as the back end language. I also used knowledge from Programming Languages I and II which were the subjects where I taught myself how to solve problems and troubleshoot during the semestral projects.

At the beginning of my practice, I had to learn the Robot Framework and everything about the robotic system which could not be taught at school or elsewhere. Since the robotic system is proprietary I had to pick up everything from scratch and the learning period was therefore longer. Another skill I was missing was software versioning in a team. This is essential for any development process and I learned the conventions reasonably promptly. The intuitive user interface of Sourcetree and the help of my colleagues sped up this process significantly.

During my practice, I automated a considerable amount of tests on HP and Xerox devices. At the beginning, as the introductory task, I worked on authentication tests which were covered in its entirety and consisted of 37 test cases. I took in basic knowledge about the robotic hand project and learned to understand other people's code. As my main task, I worked toward automating the regression testing and managed to cover almost 70% of the full testing scope of the assigned devices. This lead to an estimated 2 man-days saved every sprint (14 days). For reference, the whole robotic system is currently saving approximately 18 man-days of manual testing. At the end of my practice, I conducted knowledge sharing sessions to share information about the progress and changes within the robotic system with my colleagues.

Overall the practice was a great help towards my upcoming career in computer science. As my first experience with a job in this field, it was a useful experience and I applied my theoretical knowledge and skills from school into practice. The company was satisfied with my acquisition as an intern and offered me a full-time job as a follow-up to my practice.

References

[1] Vyletelek, Pavel. "Accounting." Ysoft Local [online], 30 Oct. 2018, wiki.ysoft.local/display/TD/Accounting. Accessed 23 Apr. 2019.

[2] Robot Framework Foundation. "Robot Framework." Robotframework/ [online], 2019, www.robotframework.org. Accessed 22 Apr. 2019.

[3] Python Software Foundation. "What Is Python? Executive Summary." Python.Org [online], 2019, www.python.org/doc/essays/blurb. Accessed 22 Apr. 2019.

[4] Atlassian. "Sourcetree - A Free Git & Mercurial Client." Atlassian [online], 2019, www.atlassian.com/software/sourcetree. Accessed 22 Apr. 2019.

[5] Atlassian. "Jira Software." Atlassian [online], 2019, www.atlassian.com/software/jira. Accessed 22 Apr. 2019.

[6] Atlassian. "Bamboo Continuous Integration and Deployment Build Server." Atlassian [online], 2019, www.atlassian.com/software/bamboo.